Open Deep Learning Student Projects in our Lab

Frank Hutter

Machine Learning Lab

University of Freiburg, Germany

General points

- Flexible scope for most projects:
  - OK for BSc theses or MSc projects/theses
- All projects are very up-to-date
  - State-of-the-art research topics
  - **Successful projects can lead to publications**
    (typically requires some more work afterwards, though)
- **Team work recommended for MSc projects**
  - 2-3 students per team (max. 4)
  - You can get a lot more done
  - Often, team work is more efficient & rewarding
- MSc theses typically only after a MSc project in our group

Main Topics for Projects

1. Modelling Data (with Neural Networks)
2. Bayesian Optimization
3. Multiple Fidelities
4. Benchmarking
5. AutoML: Architecture Search & HPO for Deep Learning
6. Optimization of Neural Networks
7. Learning to Learn

1. Modelling Data (with Neural Networks)

- Performance of scikit-learn classifiers & hyperparams
  - We would like to fit a good model to the data we collected
  - Handle conditional hyperparameters
  - Handle learning across data set (multi-task / meta-learning)
- Predicting learning curves
  - Extend work from existing MSc thesis
  - Superset of course project
- Horizon price:
  - http://ec.europa.eu/research/horizonprize/index.cfm?prize=bigdata / http://ec.europa.eu/research/horizonprize/bigdata
  - Learning from geospatial temporal data; forecasting electricity grid traffic; 6h time-limit → probably not deep learning

2. Bayesian Optimization

- Comparing an implementation of another acquisition function (called “knowledge gradient“) to our work
- Evaluate an implementation of approximate Gaussian processes (GPs) for Bayesian optimization
- In general:
  - There‘s a fairly steep learning curve in BO
  - Therefore, these two projects would focus more on evaluations & combining existing methods

3. Multiple Fidelities

- There are many ways to approximate a blackbox function
  - Initial learning curves
  - Subsample of dataset
  - Downsampled images
  - …

- So far, HB and BO-HB can only exploit either one of these
  - We'd like to exploit many of these multiple fidelities:
    e.g., short runs on a subsample of downsampled images
  - Of course, the goal is still to find a configuration that is good for long runs on all full-sized images
  - Simple approaches might do well (like Hyperband did);
    potentially could be improved by a model later (like BO-HB)

4. Benchmarking

- Benchmarking of hyperparameter optimization (HPO) or neural architecture search (NAS) algorithms is a problem
  - A benchmark is the combination of a code base and a dataset
    - But everyone just reports, e.g., their performance on CIFAR
    - The code base (which optimizer was used, with which regularization and dozens of other tricks) is a very strong confounding factor

- Solution: create a collection of benchmarks that includes both code and datasets
  - We have a preliminary version: HPOlib2
  - We also need to include multi-fidelity benchmarks
  - Use this to cleanly benchmark different HPO / NAS algorithms

4. Benchmarking

- Project: create better **surrogate benchmarks**
  - We collect a lot of performance data with different configurations
    - E.g., their entire learning curves; this is expensive
  - We then fit a model to these to create a new "surrogate" benchmark
    - Model predictions are cheap
    - These are all that is needed for running a hyperparameter optimizer
  - With these, we can benchmark hyperparameter optimizers in seconds instead of days
    - This provides a solid foundation for rigorous experiments in the field

4. Benchmarking

- Evaluating neural nets vs. other ML algos as dataset size increases
  - Is it really the case that they do scale better?
  - Is it really the case that for small data sets other algorithms perform better
    - Or do we just need to find the right regularization?

- Hyperparameter importance of neural networks
  - Which hyperparameters are most important?
  - Which hyperparameters have correlated effects?
  - What are good ranges for the hyperparameters?
  - Can we learn a mapping from dataset characteristics to good hyperparameter settings?

5. Auto-Net: NAS & HPO for Deep Learning

- Auto-Net is not yet at the level of an easy-to-use tool
  - Parameterization of a space of networks in TensorFlow

- Possible projects:
  - Implement different regularization methods / evaluate them for ParamNet
  - Study how well neural networks work on small datasets
  - Actually create an Auto-Net tool that is easy to use
  - With Bosch:
    - current NAS methods focus on classification / regression
    - extend this work to semantic segmentation

6. Optimization of Neural Networks

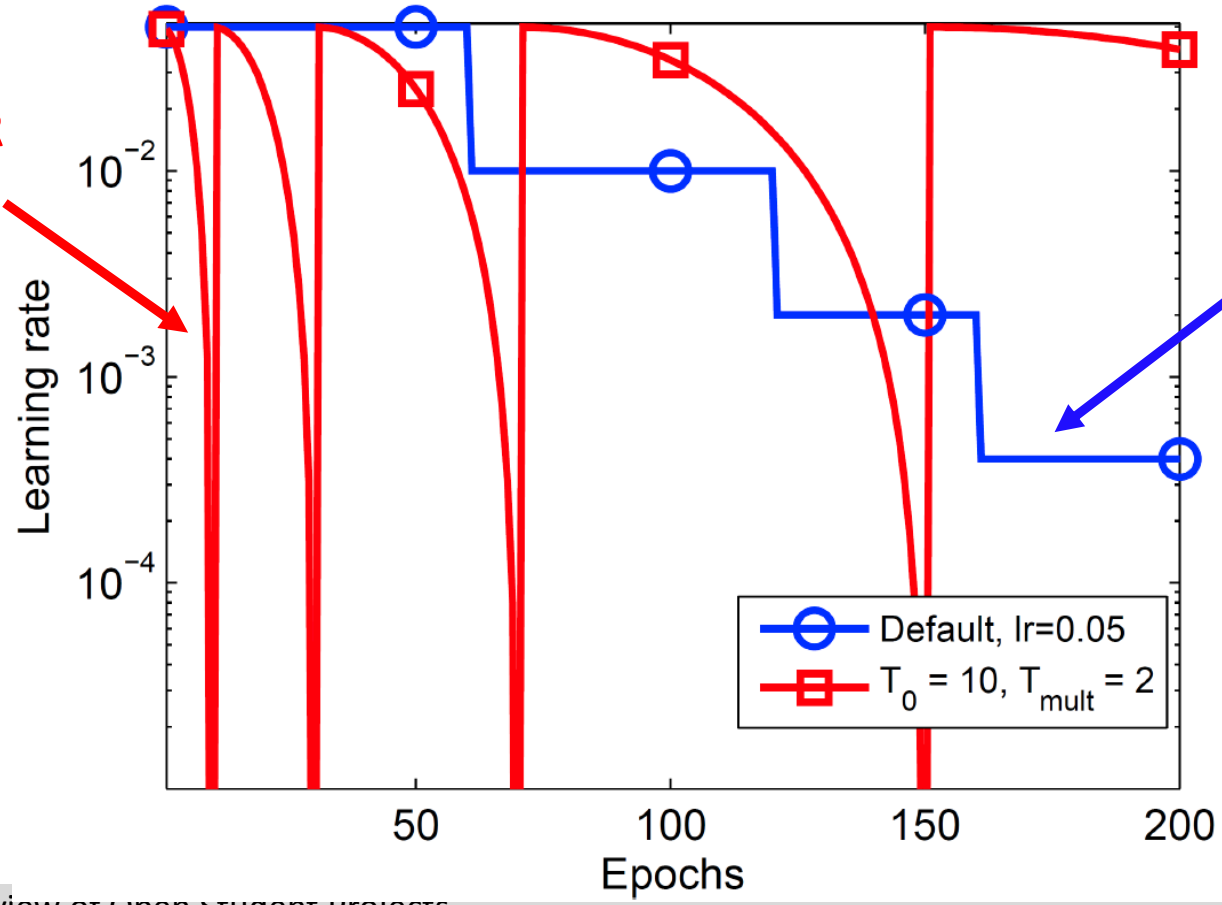
- This is a very important part of deep learning
- But there is no great solution yet; still lots of trial & error
- Better optimizers can be found manually
- It is also possible to use ML techniques to learn better optimizers

Stochastic Gradient Descent with Warm Restarts (SGDR)

[Loshchilov & Hutter, ICLR 2017]

- New Learning Rate Schedule
  - Iterative stages of convergence
  - Each stage following a **cosine annealing schedule**

New SGDR schedule

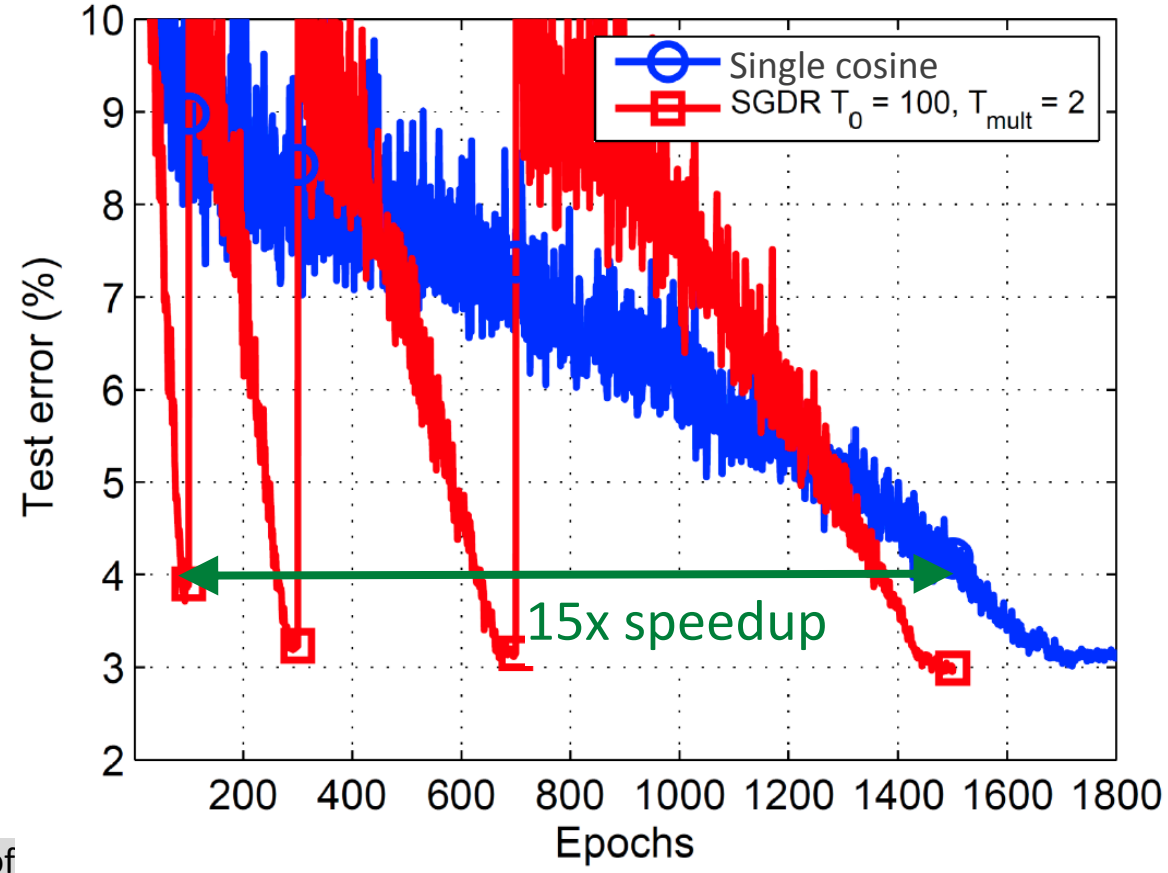


Standard stepwise learning rate schedule

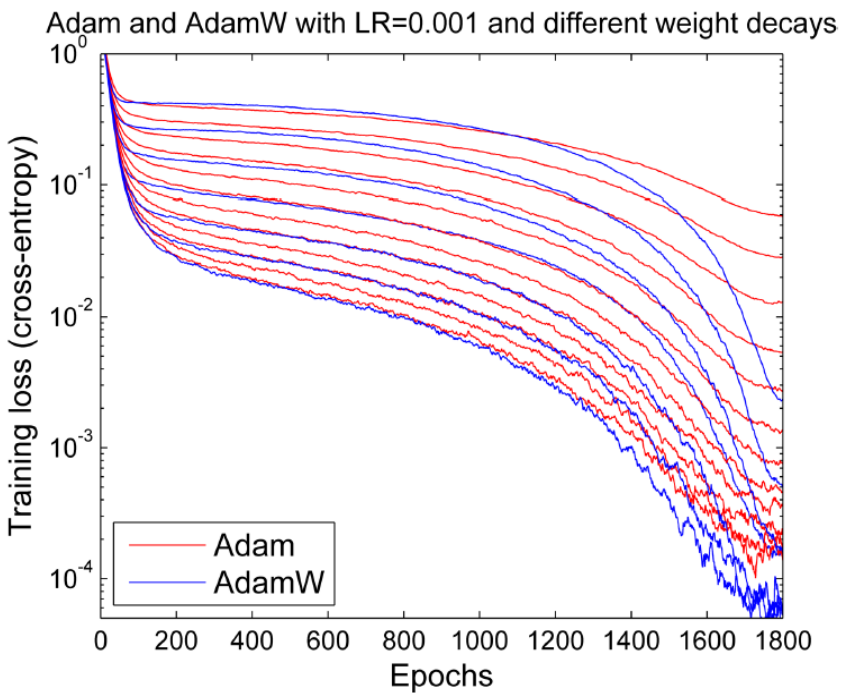
Stochastic Gradient Descent with Warm Restarts (SGDR)

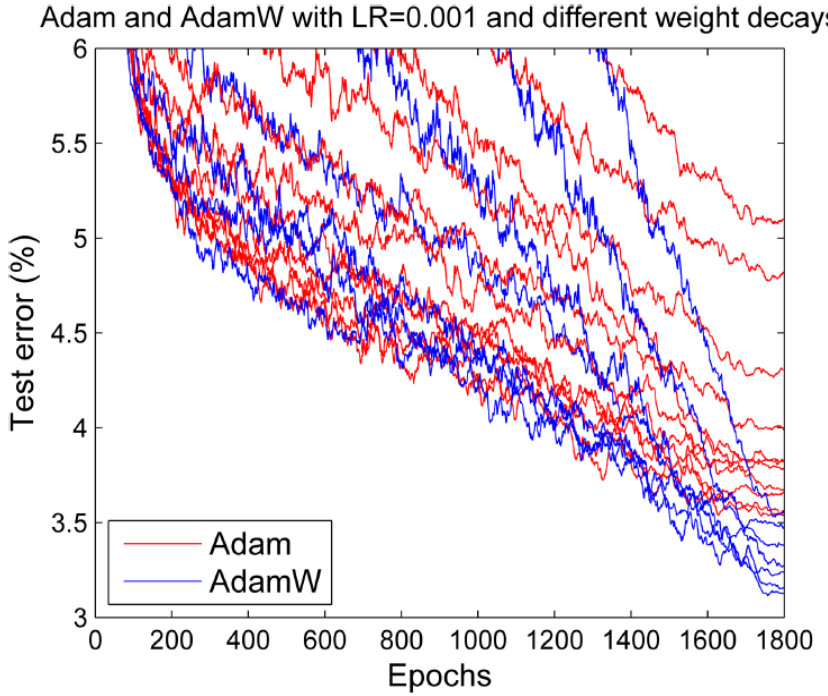
[Loshchilov & Hutter, ICLR 2017]

- Results
  - Cosine annealing is better than other learning rate schedules
  - Final performance of SGDR: similar to single cosine annealing
  - Much improved anytime performance (up to 15x faster)



- Fixing weight decay in Adam
  - Adam: most popular optimizer (> 4000 citations in 2017)
  - We exposed that in adaptive gradient algorithms, L2 regularization != weight decay
    - Standard L2 regularization does not generalize well; weight decay does
    - 20% lower error rates





Projects about Optimizers

- *Why* does cosine annealing work better than the stepwise schedule?
  - When does which schedule work best?
  - This could also involve other tasks, such as RL
- Follow-up on fixing weight decay in Adam
  - Evaluate this approach for more benchmarks
  - Evaluate using both L2 regularization and weight decay
  - Evaluate this approach for the newest versions of Adam / AMSgrad

Optimizing a policy network for Atari

- RL is the main approach for playing Atari
  - See the DQN Nature paper

- Evolutionary strategies are an alternative
  - All we need is a policy network that maps states to actions
  - We can optimize this network by evolutionary strategies
    - Highly parallelizable
    - Well studied
    - OpenAI published a paper about this last year
    - Previous MSc project: showed that even simpler methods work
    - Lots of ideas for follow-up; code base available to start from

7. Learning to Learn / Optimize

- View an optimizer (like SGD, Adam, etc) as a mapping from some state to the next step to be applied
  - Parameterize that mapping as a neural network
  - Use RL or gradient descent to optimize that network

- Project
  - Bootstrap from existing code
  - Can we learn cosine annealing & SGDR?
  - Can we apply this for different hyperparameters (weight decay, etc)?
  - Can we learn an optimizer that is actually better than Adam etc in practice?

7. Learning to Learn / Optimize

- Learning to learn without gradient descent by gradient descent
  - Learns a global blackbox optimizer
  - State: set of points in the space already evaluated, and their performance
  - Action: which point in the space to evaluate next
  - Loss: performance obtained (e.g., in fixed horizon)
  - Policy network, to be optimized with SGD

- Project: reimplement this approach
  - Code is not available

7. Learning to Learn / Optimize

- Learning an entire algorithm component
  - For basically any algorithm; here: a SAT solver
  - Represent the algorithm component as a neural net with the component's output as the network output.
  - Use policy gradient to optimize the network to improve the algorithm's overall performance
  - Expensive in terms of compute time, but (in principle) very cheap to set up in terms of human time

Main Topics for Projects

1. Modelling Data (with Neural Networks)
2. Bayesian Optimization
3. Multiple Fidelities
4. Benchmarking
5. AutoML: Architecture Search & HPO for Deep Learning
6. Optimization of Neural Networks
7. Learning to Learn